GR551x Serial Port Profile示例手册

版本： **1.9**

发布日期： **2021-08-06**

深圳市汇顶科技股份有限公司

商标声明

GOODiX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗中或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田保税区腾飞工业大厦B座2层、13层

电话：+86-755-33338828　　传真：+86-755-33338099

网址：www.goodix.com

前言

编写目的

本文档介绍如何使用和验证GR551x SDK中的Serial Port Profile（SPP）示例，旨在帮助用户快速进行二次开发。

读者对象

本文适用于以下读者：

- GR551x用户
- GR551x开发人员
- GR551x测试人员
- 开发爱好者
- 文档工程师

版本说明

本文档为第7次发布，对应的产品系列为GR551x。

修订记录

版本	日期	修订内容
1.0	2019-12-08	首次发布
1.3	2020-03-16	更新文档中页脚的时间
1.5	2020-05-30	修改“应用详解”章节中GUS_TX_NTF_ENABLE参数名称
1.6	2020-06-30	基于SDK刷新版本
1.7	2020-12-15	更新GRToolbox软件界面截图
1.8	2021-04-26	优化“初次运行”和“应用详解”章节
1.9	2021-08-06	更新“准备工作”章节

目录

1 简介

Serial Port Profile（SPP）定义了如何使用BLE技术，将虚拟串行端口中的数据透传到对端BLE设备中。

Bluetooth SIG（Bluetooth Special Interest Group，蓝牙技术联盟）未定义标准的基于BLE的串口透传Profile。为方便用户使用Goodix自定义SPP，本文将介绍如何使用以及验证GR551x SDK提供的Goodix SPP示例。

在进行操作前，可参考以下文档。

表 1-1 文档参考

名称	描述
应用及自定义GR551x Sample Service	介绍实现自定义Service的相关知识
GR551x开发者指南	GR551x软硬件介绍、快速使用及资源总览
Bluetooth Core Spec	Bluetooth官方标准核心规范
Bluetooth GATT Spec	Bluetooth Profile和Service的详细信息查看地址：http://www.bluetooth.com/specifications/gatt
J-Link用户指南	J-Link使用说明：www.segger.com/downloads/jlink/UM08001_JLink.pdf
Keil用户指南	Keil详细操作说明：www.keil.com/support/man/docs/uv4/

2 Profile概述

Goodix SPP定义了两种设备角色：

- 发起设备（Initiator）：主动发起连接请求，连接另一台设备。
- 接收设备（Acceptor）：等待其他设备的主动连接。

两者之间连接建立和数据透传的过程如图 2-1所示。

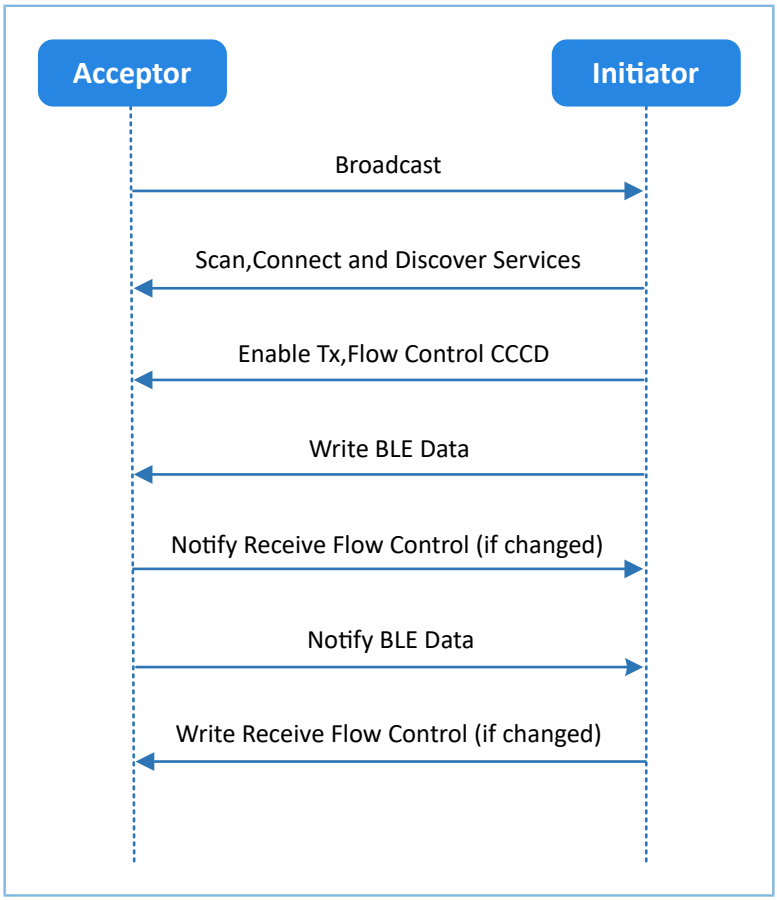

图 2-1 发起设备与接收设备交互流程图

Goodix SPP中仅定义了GR551x数据透传服务（Goodix UART Service，GUS）。该服务由Goodix自定义，专属128位UUID为A6ED0201-D344-460A-8075-B9E8EC90D71B，用于传输数据以及更新BLE数据流控制状态。

GUS包含三个特征：

- RX Characteristic：接收发起设备写入的数据。
- TX Characteristic：发送来自串口的数据至发起设备。
- Flow Control Characteristic：更新接收和发起设备的接收BLE数据能力状态（0x00：无法接收更多BLE数据；0x01：能够继续接收BLE数据）。

Characteristic的具体描述如表 2-1 所示：

表 2-1 GUS Characteristic

Characteristic	UUID	Type	Support	Security	Properties
RX	A6ED0202-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Write
TX	A6ED0203-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Notify
Flow Control	A6ED0204-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Notify, Write

3 初次运行

本章介绍如何快速验证GR551x SDK中的SPP示例。

说明:

SDK_Folder为GR551x SDK的根目录。

3.1 准备工作

验证并测试Goodix SPP示例之前，请完成以下准备工作。

- 硬件准备

表 3-1 硬件准备

名称	描述
J-Link工具	SEGGER公司推出的JTAG仿真器，如需更多了解，请访问 www.segger.com/products/debug-probes/j-link/
开发板	GR5515 Starter Kit开发板（以下简称开发板）
数据线	Micro USB 2.0数据线

- 软件准备

表 3-2 软件准备

名称	描述
Windows	Windows 7/Windows 10操作系统
J-Link Driver	J-Link驱动程序，下载网址：www.segger.com/downloads/jlink/
Keil MDK5	IDE工具，支持MDK-ARM 5.20 及以上版本，下载网址：www.keil.com/download/product/
GRToolbox（Android）	BLE调试工具，位于SDK_Folder\tools\GRToolbox
GRUart（Windows）	串口调试工具，位于SDK_Folder\tools\GRUart
GProgrammer（Windows）	Programming工具，位于SDK_Folder\tools\GProgrammer

3.2 固件烧录

SPP示例工程的源码位于SDK_Folder\projects\ble\ble_peripheral\ble_app_uart。

用户可通过GProgrammer将SPP示例的*ble_app_uart_fw.bin*固件烧录至开发板。GProgrammer烧录固件的具体操作方法，请参考《GProgrammer用户手册》。

说明:

- *ble_app_uart_fw.bin*位于SDK_Folder\projects\ble\ble_peripheral\ble_app_uart\build。
- GProgrammer位于SDK_Folder\tools\GProgrammer。

3.3 测试验证

准备好开发板、GRToolbox和GRUart后即可开始测试，测试步骤如下：

1. 通过GRToolbox与开发板建立连接。

   启动手机GRToolbox扫描设备，发现广播名为“Goodix_UART”的设备（广播名可在*user_app.c*文件中进行修改）。

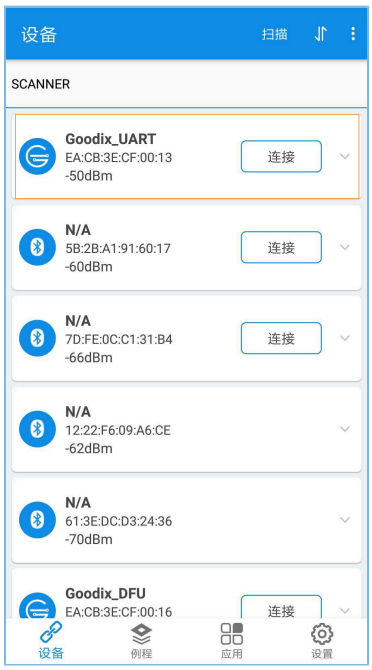

图 3-1 手机端扫描到“Goodix_UART”设备

说明:

本文中GRToolbox的截图仅供用户了解操作步骤，实际界面请参考最新版本GRtoolbox。

点击“Goodix_UART > 连接”，手机界面显示Goodix UART Service相关信息，包括TX Characteristic、RX Characteristic和Flow Control Characteristic，如图 3-2所示。

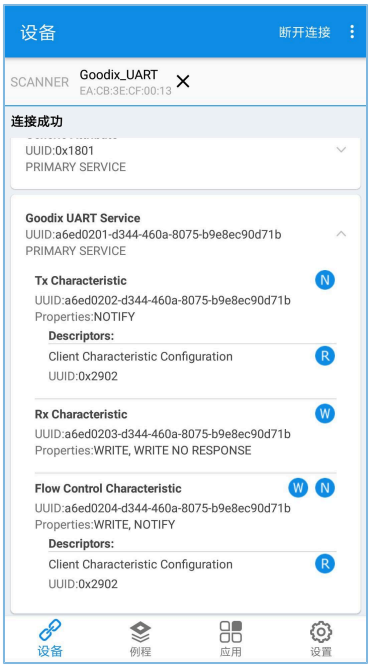

图 3-2 在手机端发现Goodix UART Service

2. 采用GRToolbox发送数据。

在GRToolbox中使能对端设备上GUS的TX Characteristic通知（Notify）和Flow Control Characteristic通知（Notify），完成后如图 3-3所示：

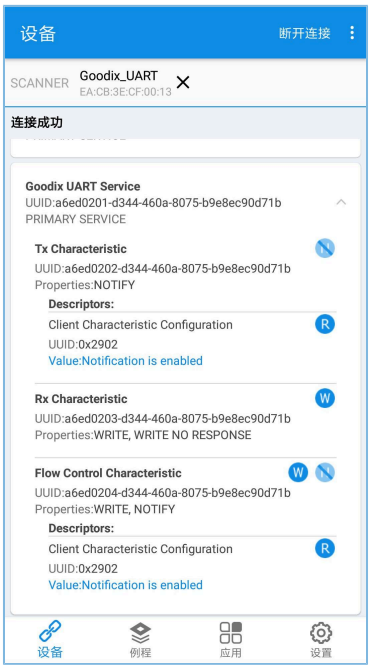

图 3-3 使能TX Characteristic通知和Flow Control Characteristic通知

向对端RX Characteristic写入数据，例如输入“12345678”，点击“发送”。

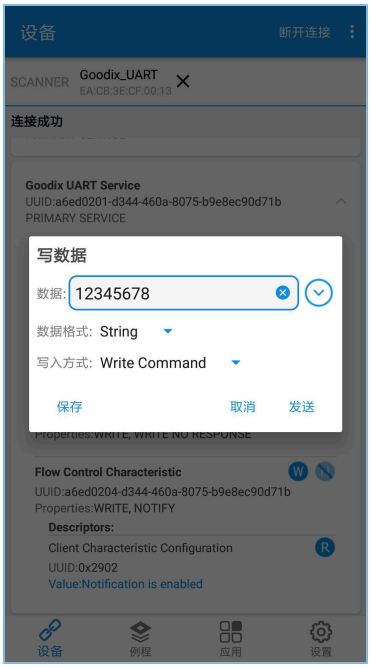

图 3-4 输入RX特征值

GRUart的“Receive Data”区域将显示GRToolbox发送的数据，如图 3-5所示。

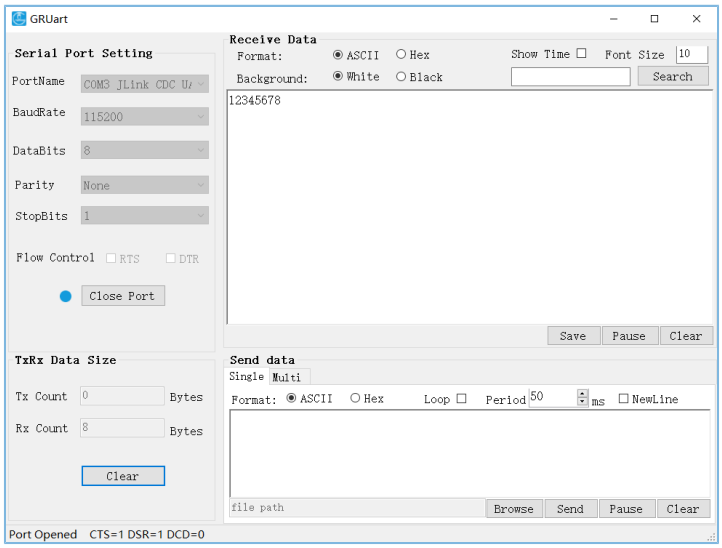

图 3-5 GRUart中显示GRToolbox发送的数据

3. 采用GRUart发送数据。

   在GRUart的“Send data”区域输入“abcdefgh”，点击“Send”。

   在GRToolbox中，TX Characteristic的Value将显示GRUart发送的数据，如图 3-6所示。

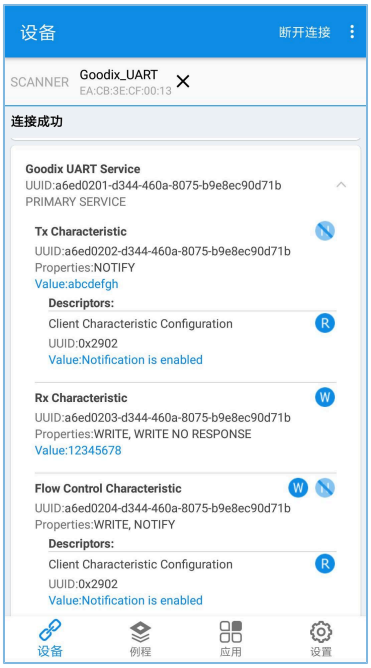

图 3-6 GRToolbox中显示GRUart发送的值

若实际情况符合上述说明，则Goodix SPP示例运行成功。

4 应用详解

本章将介绍Goodix SPP示例的运行流程和关键代码。

4.1 运行流程

Goodix SPP示例被发起设备扫描、连接之后，主要流程如图 4-1所示：

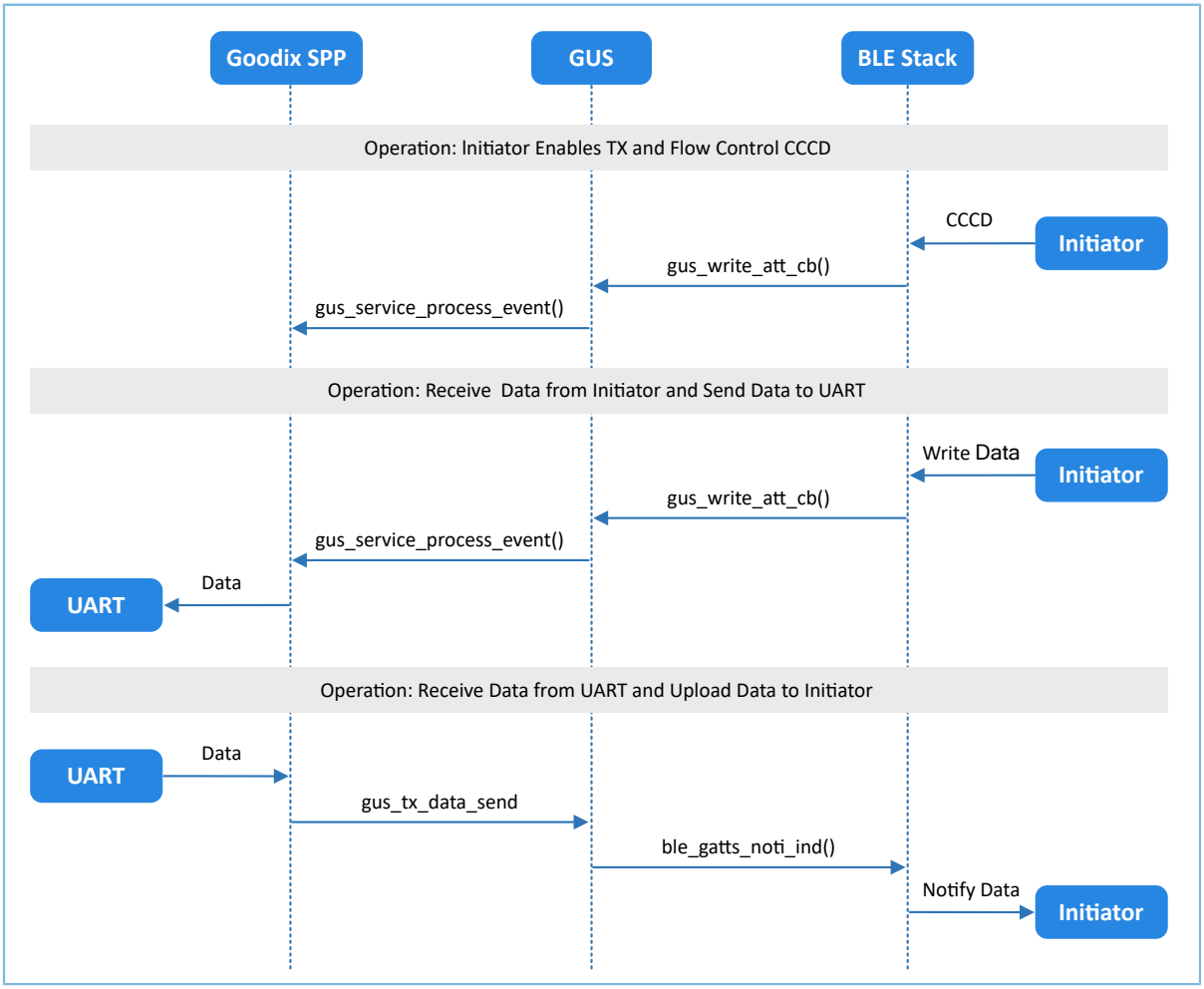

图 4-1 Goodix SPP示例主要流程

4.2 关键代码

下文介绍了发起设备与接收设备交互过程中的关键代码。

4.2.1 开启数据发送特性和数据流控特性通知

当发起设备发出开启接收设备上GUS的TX Characteristic通知的指令后，GUS解析该指令以“GUS_EVT_TX_PORT_OPENED”事件上报至应用层，开启TX Characteristic的通知。此时接收设备可以将来自串口的数据传输到发起设备。

当发起设备发出开启接收设备上GUS的Flow Control Characteristic通知的指令后，GUS解析该指令并以“GUS_EVT_FLOW_CTRL_ENABLE”事件上报至应用层，开启Flow Control Characteristic的通知。此时接收设备可以将接收BLE数据能力的状态通知到发起设备。

路径：工程目录下的user_app\user_app.c

名称：gus_service_process_event();

```
static void gus_service_process_event(gus_evt_t *p_evt)
{
    switch (p_evt->evt_type)
    {
        case GUS_EVT_TX_PORT_OPENED:
            transport_flag_set(GUS_TX_NTF_ENABLE, true);
            break;
        case GUS_EVT_FLOW_CTRL_ENABLE:
            transport_flag_set(BLE_FLOW_CTRL_ENABLE, true);
            break;
        ...
    }
}
```

4.2.2 接收数据并发送至串口

当接收设备接收到来自发起设备的BLE数据后，GUS以“GUS_EVT_RX_DATA_RECEIVED”事件上报至应用层，应用层调用ble_to_uart_push()把数据存放在对应环形缓存区中。

路径：工程目录下的user_app\user_app.c

名称：gus_service_process_event();

```
static void gus_service_process_event(gus_evt_t *p_evt)
{
    switch (p_evt->evt_type)
    {
         ...
         case GUS_EVT_TX_DATA_RECEIVED:
                ble_to_uart_push(p_evt->p_data, p_evt->length);
                break;
         ...
    }
}
```

transport_schedule()函数运行在main()的while循环中，负责执行轮询环形缓存区的任务。若其检测到环形缓存区存有新的数据，则调用transport_uart_data_send()从环形缓冲区取出数据并将数据发送至串口。

路径：工程目录下的user_app\transport_scheduler.c

名称：transport_uart_data_send()

```
static void transport_uart_data_send(void)
{
```

```
    uint16_t read_len;
    uint16_t items_avail;

    items_avail = ring_buffer_items_count_get(&s_ble_RX_ring_buffer);

    if (items_avail > 0)
    {
        read_len = ring_buffer_read(&s_ble_RX_ring_buffer, s_uart_TX_data,
                                    UART_ONCE_SEND_SIZE);
        transport_flag_set(UART_TX_CPLT, false);
        uart_TX_data_send(s_uart_TX_data, read_len);
    }
}
```

4.2.3 接收串口数据并传输至发起设备

接收设备从串口接收完数据后，将在串口事件处理函数app_uart_evt_handler()中暂存数据至环形缓存区。

路径：工程目录下的user_platform\user_periph_setup.c

名称：app_uart_evt_handler();

```
static void app_uart_evt_handler(app_uart_evt_t *p_evt)
{
    if (APP_UART_EVT_RX_DATA == p_evt->type)
    {
        uart_to_ble_push(s_uart_RX_buffer, p_evt->data.size);
        app_uart_receive_async(APP_UART_ID, s_uart_RX_buffer,
                               UART_RX_BUFFER_SIZE);
     }
     ...
}
```

当没有BLE数据发送任务时，transport_schedule()函数将调用transport_ble_data_send()函数轮询环形缓存区。如果环形缓冲区存有待传输数据，则执行BLE数据发送任务。

路径：工程目录下的user_app\transport_scheduler.c

名称：transport_ble_data_send();

```
static void transport_ble_data_send(void)
{
    uint16_t read_len;
    uint16_t items_avail;

    items_avail = ring_buffer_items_count_get(&s_uart_RX_ring_buffer);

    if (items_avail > 0)
    {
        read_len = ring_buffer_read(&s_uart_RX_ring_buffer, s_ble_TX_data,
                                    s_mtu_size - 3);
        transport_flag_set(BLE_TX_CPLT, false);
```

```
        gus_TX_data_send(0, s_ble_TX_data, read_len);
    }
}
```

当一次BLE数据发送完成后，GUS将向应用层上报“GUS_EVT_TX_DATA_SENT”事件，应用层调用transport_ble_continue_send()函数查询环形缓存区。如果环形缓冲区还存有待传输数据，则继续取出数据传输至发起设备。

路径：工程目录下的user_app\transport_scheduler.c

名称：transport_ble_continue_send();

```
void transport_ble_continue_send(void)
{
    ...
    transport_flag_set(BLE_SCHEDULE_ON, true);
    // Read data from m_uart_RX_ring_buffer and send to peer via BLE.
    if (transport_flag_cfm(BLE_TX_FLOW_ON))
    {
        items_avail = ring_buffer_items_count_get(&s_uart_RX_ring_buffer);

        if (items_avail > 0)
        {
            read_len = ring_buffer_read(&s_uart_RX_ring_buffer, s_ble_TX_data,
                                        s_mtu_size - 3);
            transport_flag_set(BLE_TX_CPLT, false);
            transport_flag_set(BLE_SCHEDULE_ON, false);
            gus_TX_data_send(0, s_ble_TX_data, read_len);
        }
    }
}
```

5 常见问题

本章描述了在使用Goodix SPP示例时，可能出现的问题、原因及处理方法。

5.1 手机多次收到小于等于20 bytes的数据

- 问题描述

  当通过GRUart串口助手输入数据长度超过20 bytes时，手机会分若干次接收到数据。

- 问题分析

  在双方未进行最大数据单元（MTU）更改交换时，都使用默认MTU为23 bytes，其中操作码为1 byte，属性句柄为2 bytes，因此，单次发送数据长度为20 bytes。

  当需要发送至手机端数据超过20 bytes时，会拆分为若干小于等于20 bytes字节有序发送。

  可通过更改MTU值的方式解决。

- 处理方法

  在GRToolbox右上角，点击“⋮ > 设置最大数据单元”，如图 5-1所示。

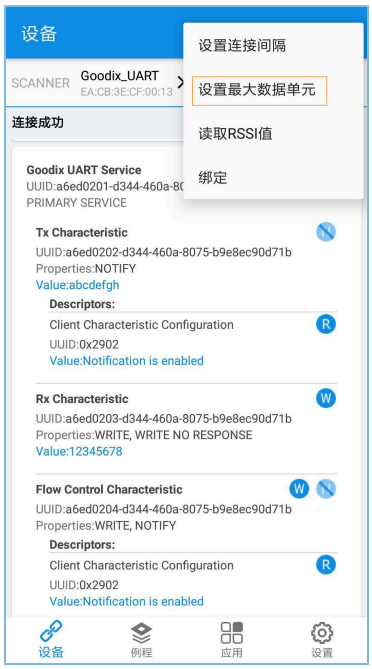

图 5-1 选择“设置最大数据单元”

说明:

MTU在一次连接中仅能更新一次，如果当前没有更新生效，则可能是因为之前已经发生过一次MTU更新事件。

输入自定义MTU值，如“400”bytes，并点击“确定”进行更新（MTU的取值范围为23 ～ 512 bytes）。

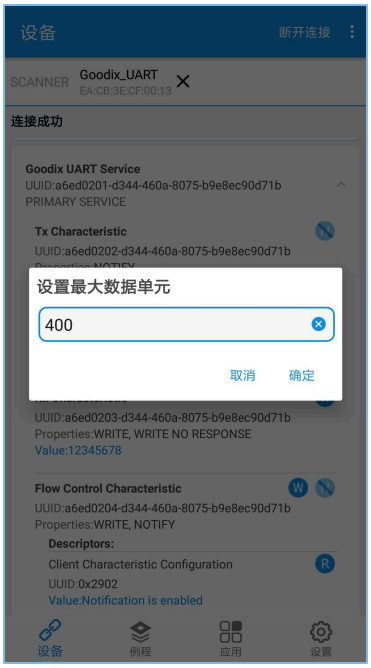

图 5-2 设置最大数据单元

5.2 数据发送为字符串但接收为十六进制

- 问题描述

  串口发送数据为字符串，如“abcdefgh”，但GRToolbox接收数据显示为十六进制（字节）数据。

- 问题分析

  未设置正确的数据显示格式。

- 处理方法

  GRToolbox在数据（包括接收数据和发送数据）显示时，均可设置数据显示格式为字符串和字节。如图 5-3所示，接收数据显示为字节格式。

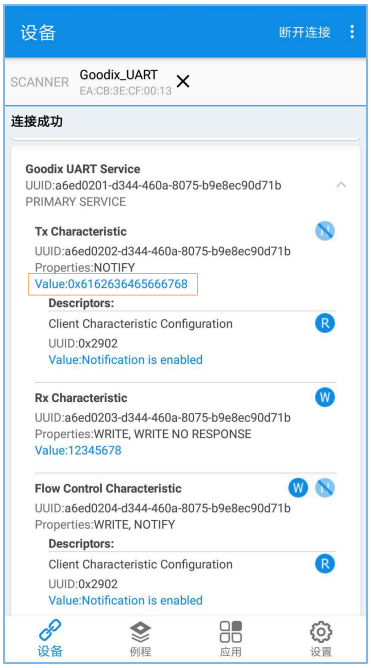

图 5-3 接收数据字节格式显示

此时，点击“Value”，即可弹出数据显示格式选择菜单。

图 5-4 选择数据显示格式

选择数据格式为“ASCII”后，点击“确定”，则显示为字符串“abcdefgh”，如图 5-5所示。

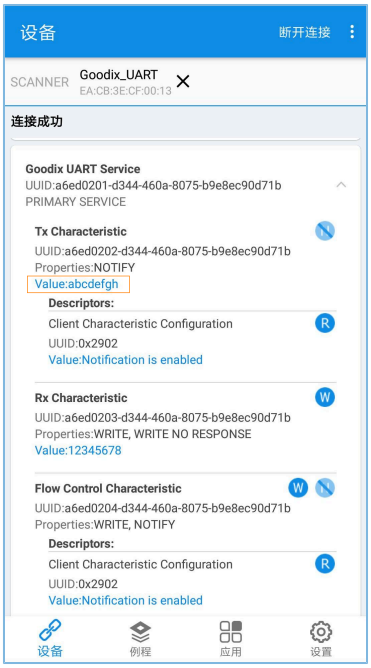

图 **5-5** 显示为字符串“abcdefgh”

6 附录：吞吐率测试结果

基于开发板，使用Goodix SPP示例进行了BLE吞吐率测试。

以下测试结果包括：115200 bps、230400 bps、460800 bps串口波特率，以及1 M、2 M PHY不同条件下，不同透传模式下的吞吐率。

表 6-1 不同模式下的吞吐率

波特率（bps）	透传模式	1M PHY	2M PHY
115200	接收设备 → 发起设备	10.032 KB/s	10.246 KB/s
	接收设备 ← 发起设备	10.015 KB/s	10.167 KB/s
	接收设备 ↔ 发起设备	19.534 KB/s	19.758 KB/s
230400	接收设备 → 发起设备	20.329 KB/s	21.011 KB/s
	接收设备 ← 发起设备	20.009 KB/s	19.907 KB/s
	接收设备 ↔ 发起设备	40.069 KB/s	41.01 KB/s
460800	接收设备 → 发起设备	37.38 KB/s	37.826 KB/s
	接收设备 ← 发起设备	37.38 KB/s	37.648 KB/s
	接收设备 ↔ 发起设备	70.243 KB/s	71.141 KB/s